

IN THE CLAIMS:

- 1 1. (Currently Amended) A method for executing uniprocessor (UP) coded workloads in a multiprocessor (MP) computer system without having to rewrite the UP-coded workloads' code, the method comprising the steps:
 - 4 organizing the UP-coded workloads into one or more concurrency groups, wherein UP-coded workloads in the same concurrency group are not permitted to execute concurrently with one another in the MP computer system;
 - 7 scheduling first and second execution vehicles that respectively execute on different processors in the MP computer system at substantially the same time;
 - 9 acquiring a first concurrency group by the first execution vehicle and a second concurrency group by the second execution vehicle; and
 - 11 executing UP-coded workloads in the first concurrency group through the first execution vehicle at substantially the same time as UP-coded workloads in the second concurrency group are executed through the second execution vehicle.
- 1 2. (Original) The method according to claim 1, wherein the UP-coded workloads are UP-coded threads, and the first and second execution vehicles are first and second processes.
- 1 3. (Original) The method according to claim 1, wherein the UP-coded workloads are messages, and the first and second execution vehicles are first and second threads.
- 1 4. (Original) The method according to claim 1, wherein the step of acquiring the first and second concurrency groups further comprises:
 - 3 dequeuing from a concurrency-group run queue a first concurrency-group data structure associated with the first concurrency group; and

5 dequeuing from the concurrency-group run queue a second concurrency-group
6 data structure associated with the second concurrency group.

1 5. (Original) The method according to claim 4, further comprising:
2 setting a first CG flag in the first concurrency-group data structure to a value indicating that the first concurrency group is in a running state; and
3 setting a second CG flag in the second concurrency-group data structure to a value indicating that the second concurrency group is in a running state.

1 6. (Original) The method according to claim 4, further comprising:
2 appending UP-coded workloads enqueued on a first current queue in the first concurrency-group data structure onto a first active queue in the first concurrency-group data structure; and
3 appending UP-coded workloads enqueued on a second current queue in the second concurrency-group data structure onto a second active queue in the second concurrency-group data structure.

1 7. (Original) The method according to claim 6, further comprising:
2 dequeuing UP-coded workloads in the first and second concurrency groups from the first and second active queues, respectively; and
3 executing the dequeued UP-coded workloads to completion.

1 8. (Original) The method according to claim 5, further comprising:
2 in response to the first execution vehicle finishing execution of the UP-coded workloads in the first concurrency group, the first execution vehicle performing the steps:
3 A) if at least one UP-coded workload in the first concurrency group is executable:
4 (i) setting the value of the first CG flag to a value indicating that the first concurrency group is in a queued state;

(ii) re-enqueueing the first concurrency-group data struc-

ture onto the concurrency-group run queue;

10 B) if there are not any UP-coded workloads in the first concurrency
11 group that are executable, setting the first CG flag to a value indicating that the
12 first concurrency group is in a suspended state;

13 C) dequeuing from the concurrency-group run queue a third concurrency-
14 group data structure associated with a third concurrency group; and

15 D) setting a third CG flag in the third concurrency-group data structure to
16 a value indicating that the third concurrency group is in a running state.

1 9. (Original) The method according to claim 1, wherein at least one of the UP-coded
2 workloads is organized into the one or more concurrency groups at run-time.

1 10. (Original) The method according to claim 1, wherein the MP computer system is a
2 network cache.

1 11. (Original) A multiprocessor (MP) computer system configured to execute uniproces-
2 sor (UP) coded threads without having to rewrite the UP-coded threads' code, the MP
3 computer system comprising:

4 a plurality of processors;

5 a memory having a plurality of storage locations addressable by the plurality of
6 processors for storing data and program code, the memory being configured to store a
7 separate concurrency-group data structure for each of a plurality of concurrency groups
8 each concurrency-group data structure comprising:

9 an active-queue pointer storing a location in the memory of an active
10 queue of UP-coded thread messages associated with UP-coded threads in an ex-
11 ecutable state; and

12 a current-queue pointer storing a location in the memory of a current
13 queue of UP-coded thread messages associated with UP-coded threads waiting to
14 be transferred to the active queue.

1 12. (Original) The MP computer system according to claim 11, wherein each concurrency-group data structure further comprises a CG flag that stores a value indicating an
2 operational state of a concurrency group associated with the concurrency-group data
3 structure.

1 13. (Original) The MP computer system according to claim 11, wherein each UP-coded
2 thread message stored in the active queue and current queue stores a location in the
3 memory of a top of a call stack associated with a specific UP-coded thread.

1 14. (Original) The MP computer system according to claim 13, wherein the call stack is
2 accessible through a thread control block (TCB) associated with the specific UP-coded
3 thread, the TCB including a CG pointer for storing a memory location of a concurrency-
4 group data structure.

1 15. (Original) The MP computer system according to claim 11, wherein each concurrency-group data structure further comprises meta-data information associated with a
2 concurrency group.

1 16. (Original) The MP computer system according to claim 11, wherein the MP computer
2 system is a network cache.

1 17. (Currently Amended) An apparatus for executing uniprocessor (UP) coded workloads
2 in a multiprocessor (MP) computer system without having to rewrite the UP-coded work-
3 loads' code, the method comprising the steps:

4 means for organizing the UP-coded workloads into one or more concurrency
5 groups, wherein UP-coded workloads in the same concurrency group are not permitted to
6 execute concurrently with one another in the MP computer system;

7 means for scheduling first and second execution vehicles that respectively execute
8 on different processors in the MP computer system at substantially the same time;

9 means for acquiring a first concurrency group by the first execution vehicle;

10 means for acquiring a second concurrency group by the second execution vehicle;

11 and

12 means for executing UP-coded workloads in the first concurrency group through
13 the first execution vehicle at substantially the same time as UP-coded workloads in the
14 second concurrency group are executed through the second execution vehicle.

1 18. (Original) The apparatus according to claim 17, wherein the UP-coded workloads are
2 UP-coded threads, and the first and second execution vehicles are first and second proc-
3 esses.

1 19. (Original) The apparatus according to claim 17, wherein the UP-coded workloads are
2 messages, and the first and second execution vehicles are first and second threads.

1 20. (Original) The apparatus according to claim 17, further comprising:

2 means for dequeuing from a concurrency-group run queue a first concurrency-
3 group data structure associated with the first concurrency group; and

4 means for dequeuing from the concurrency-group run queue a second concur-
5 rency-group data structure associated with the second concurrency group.

1 21. (Original) The apparatus according to claim 20, further comprising:

2 means for setting a first CG flag in the first concurrency-group data structure to a
3 value indicating that the first concurrency group is in a running state; and

4 means for setting a second CG flag in the second concurrency-group data struc-
5 ture to a value indicating that the second concurrency group is in a running state.

1 22. (Original) The apparatus according to claim 20, further comprising:

2 means for appending UP-coded workloads enqueued on a first current queue in
3 the first concurrency-group data structure onto a first active queue in the first concur-
4 rency-group data structure; and

5 means for appending UP-coded workloads enqueued on a second current queue in
6 the second concurrency-group data structure onto a second active queue in the second
7 concurrency-group data structure.

1 23. (Original) The apparatus according to claim 22, further comprising:

2 means for dequeuing UP-coded workloads in the first and second concurrency
3 groups from the first and second active queues, respectively; and

4 means for executing the dequeued UP-coded workloads to completion.

1 24. (Original) The apparatus according to claim 21, further comprising:

2 means for setting the value of the first CG flag to a value indicating that the first
3 concurrency group is in a queued state or in a suspended state; and

4 means for re-enqueueing the first concurrency-group data structure onto the con-
5 currency-group run queue.

1 25. (Currently Amended) A computer-readable media comprising instructions for execu-
2 tion in one or more processors for executing uniprocessor (UP) coded workloads in a
3 multiprocessor (MP) computer system without having to rewrite the UP-coded work-
4 loads' code, ~~the method comprising the steps:~~

5 organizing the UP-coded workloads into one or more concurrency groups,
6 wherein UP-coded workloads in the same concurrency group are not permitted to execute
7 concurrently with one another in the MP computer system;

8 scheduling first and second execution vehicles that respectively execute on differ-
9 ent processors in the MP computer system at substantially the same time;
10 acquiring a first concurrency group by the first execution vehicle and a second
11 concurrency group by the second execution vehicle; and
12 executing UP-coded workloads in the first concurrency group through the first
13 execution vehicle at substantially the same time as UP-coded workloads in the second
14 concurrency group are executed through the second execution vehicle.

1 26. (Original) The computer-readable media according to claim 25, wherein the UP-
2 coded workloads are UP-coded threads, and the first and second execution vehicles are
3 first and second processes.

1 27. (Original) The computer-readable media according to claim 25, wherein the UP-
2 coded workloads are messages, and the first and second execution vehicles are first and
3 second threads.

1 28. (Currently Amended) A method for executing workloads in a multiprocessor (MP)
2 computer system, ~~the method comprising the steps:~~

3 organizing the workloads into one or more concurrency groups, wherein work-
4 loads in the same concurrency group are not permitted to execute concurrently with one
5 another in the MP computer system;

6 scheduling first and second execution vehicles that respectively execute on differ-
7 ent processors in the MP computer system at substantially the same time;

8 acquiring a first concurrency group by the first execution vehicle and a second
9 concurrency group by the second execution vehicle; and

10 executing workloads in the first concurrency group through the first execution ve-
11 hicle at substantially the same time as workloads in the second concurrency group are
12 executed through the second execution vehicle.

1 29. (Original) The method according to claim 28, wherein the step of acquiring the first
2 and second concurrency groups further comprises:

3 dequeuing from a concurrency-group run queue a first concurrency-group data
4 structure associated with the first concurrency group; and

5 dequeuing from the concurrency-group run queue a second concurrency-group
6 data structure associated with the second concurrency group.

1 30. (Original) The method according to claim 29, further comprising:

2 setting a first CG flag in the first concurrency-group data structure to a value indi-
3 cating that the first concurrency group is in a running state; and

4 setting a second CG flag in the second concurrency-group data structure to a
5 value indicating that the second concurrency group is in a running state.

1 31. (Original) The method according to claim 29, further comprising:

2 appending workloads enqueued on a first current queue in the first concurrency-
3 group data structure onto a first active queue in the first concurrency-group data struc-
4 ture; and

5 appending workloads enqueued on a second current queue in the second concur-
6 rency-group data structure onto a second active queue in the second concurrency-group
7 data structure.

1 32. (Original) The method according to claim 31, further comprising:

2 dequeuing workloads in the first and second concurrency groups from the first
3 and second active queues, respectively; and

4 executing the dequeued workloads to completion.

1 33. (Original) The method according to claim 30, further comprising:

2 in response to the first execution vehicle finishing execution of the workloads in
3 the first concurrency group, the first execution vehicle performing the steps:

- 4 A) if at least one workload in the first concurrency group is executable:
- 5 (i) setting the value of the first CG flag to a value indicat-
- 6 ing that the first concurrency group is in a queued state;
- 7 (ii) re-enqueueing the first concurrency-group data struc-
- 8 ture onto the concurrency-group run queue;
- 9 B) if there are not any workloads in the first concurrency group that are
- 10 executable, setting the first CG flag to a value indicating that the first concurrency
- 11 group is in a suspended state;
- 12 C) dequeuing from the concurrency-group run queue a third concur-
- 13 rency-group data structure associated with a third concurrency group; and
- 14 D) setting a third CG flag in the third concurrency-group data structure to
- 15 a value indicating that the third concurrency group is in a running state.

1 Please add new claims 34 *et al.*

1 34. (New) A method, comprising:

2 organizing a plurality of workloads into a plurality of concurrency groups,
3 wherein each workload in the same concurrency group are not permitted to execute con-
4 currently with another workload in a microprocessor (MP) computer system;

5 scheduling a plurality of execution vehicles that respectively execute on different
6 processors in the MP computer system at substantially the same time;

7 acquiring by each execution vehicle of the plurality of execution vehicles a con-
8 currency group from the plurality of concurrency groups; and

9 executing workloads in the plurality of concurrency groups through the plurality
10 of execution vehicles at substantially the same time.

1 35. (New) The method according to claim 34, wherein the workloads are uniprocessor
2 (UP) coded threads, and the plurality of vehicles are processes.

1 36. (New) The method according to claim 34, wherein the workloads are messages, and
2 the plurality of vehicles are first and second threads.

1 37. (New) The method according to claim 34, wherein the workloads are uniprocessor
2 (UP) coded workloads.